NASA Technical Memorandum 107402, Part 2
IEEE–155NO897–5000

# Software Design Improvements

## Part 2: Software Quality and the Design and Inspection Process

Vincent R. Lalli
*Lewis Research Center*
*Cleveland, Ohio*

Michael H. Packard
*Raytheon Engineers and Constructors*
*Brook Park, Ohio*

Tom Ziemianski
*Texas, Instruments Inc.*
*Dallas, Texas*

Prepared for
The International Symposium on Product Quality and Integrity
cosponsored by AIAA, ASQC/RD, ASQC/ED, IEEE/RS, IES, IIE,
SAE, SOLE, SRE, and SSS
Philadelphia, Pennsylvania, January 13–16, 1997


National Aeronautics and
Space Administration

SOFTWARE DESIGN IMPROVEMENTS
PART II -- SOFTWARE QUALITY AND THE DESIGN AND INSPECTION PROCESS

## *1. SOFTWARE DEVELOPMENT SPECIFICATIONS*

Fig 1. **IMPROVING SOFTWARE**

| |
|---|
| **-Improving software with standards & controls includes:** |
| **-Robust design -- making software fault tolerant** |
| **-Process controls -- standardizing the software development process.** |
| **-Design standards -- standardizing the software specifications** |
| **-Inspection -- standardize the software requirements inspection process.** |
| **-Inspection of Code -- standardize the software code inspection process** |

Precise and easily readable documentation and specifications are needed for a successful software project. Ideally, formal methods and specifications language should be used. Once they are written they must be understood and adhered to. To do this successfully, there must be team participation in document and specification generation. There also must be real support of the specifications, document and the verification of conformance and validation of the software itself by upper management and the team.

Fig 2. **SOFTWARE DEVELOPMENT SPECS.**

| |
|---|
| **•Software Management Plan** |
| **•Software Design Specifications** |
| **•Software Development Plan** |
| **•Plan for Formal Inspection of Software** |
| **•Software Safety Program Plan** |
| **•Software Maintenance Plan** |
| **•Configuration Management Plan** |
| **•Interface Control Document(s)** |
| **•Failure Review Boards** |
| **•Lessons Learned** |

Some of these documents and related practices should include:

(1) *A formal software management plan* that includes the software development cycle, the configuration management plan, approval authority and group charter and responsibilities. This plan would specify what other documentation is required, how interfaces are to be controlled and what the quality assurance and verification requirements are.

(2) *A formal software design specification* which includes architecture specifications and hardware interfaces.

(3) *A software development plan* that describes development activities, facilities and personnel, activity flow and the development tools used to generate the software.

(4) *A plan for formal inspection of software* that included a software quality assurance plan to integrate hardware and software safety, quality and reliability. This would have a software verification test specification and a software fault tolerance and failure modes and effects analysis specification.

(5) *A software safety program plan* which includes a software safety handbook and reliability practices specifications.

(6) *A formal plan for maintenance* and operation.

(7) *Configuration management* and documentation plans should specify recording all changes to software and the reasons for the changes. Records should include designs change that require software modifications. Also, any change in the functional capabilities, performance specifications or allocation of software to components or interfaces should be noted.

(8) *Interface control documentation* should specify linking hardware and software, and vendor-supplied software and internally generated software.

(9) *Failure review boards* are needed to review bugs, the bug removal process, and review their overall effect on the system.

(10) *Lessons learned* should be used to document problems and solutions to eliminate repetition of errors.

(11) *Test plans* that will to the greatest extent possible, validate the software system.

Once these documents are developed and procedures set up, they must be implemented, enforced and maintained. A software system safety working team (multidisciplined) can assist software engineering and continually monitor adherence to the documentation. They also have to engender respect for the need to follow the specification, not mandate them and walk away. Therefore, the team and software engineering management also needs to educate programmers in the understanding and use of the specifications.

## *2. SPECIFICATIONS & PROGRAMMING STANDARDS*

Structured programming with a well-defined design approach, extensive commenting benefits the software design

process. Standardizing formats, nomenclature and language as well as standardized compilers and platforms for the software contribute to project success as well. Besides many excellent internal company standards for software development, a number of documents exist to help in the standardization process and to gauge the maturity of the software development effort.

Fig 3. **SPECIFICATIONS & PROGRAMMING STDS.**

- **Capability Maturity Model**
- **ISO 9000-3 Software Guidelines**
- **IEEE Software Engineering Standards Collections**
- **NASA developed software standards**
- **DOD Standards.**

Some of these documents are:

(1) *The Software Engineering Institute (SEI) Capability Maturity Model (CMM)* is a method for assessing the software engineering capabilities of development organizations. It evaluates the level of process control and methodology in developing software. It is designed to rank the "maturity" of the company and its ability to undertake major software development projects.

(2) *ISO 9000-3 Software Guidelines,* Part 3, Guidelines for the application of ISO 9001 to the development, supply and maintenance of software is intended to provide suggested controls and methods.

(3) *IEEE Software Engineering Standards Collections* include 22 standards (1993 edition) covering terminology, quality assurance plans, configuration management, test documentation, requirements specifications, maintenance, metrics and other subjects.

(4) *NASA developed software standards* include NSS 1740.13, INTERIM, June 1994, NASA Software Safety Standards that expands on the requirements of NASA Management Instruction (NMI) 2410.10, NASA Software Management Assurance and Engineering Policy. These documents contain a detailed reference document list.

(5) *DOD Standards* include MIL-STD-882C, System Safety Program Requirements, DOD-STD-2167A, Defense System Software Development, MIL-STD-498, Software Development and Documentation and numerous other standards and guidelines.

## *3. NASA SOFTWARE INSPECTION ACTIVITIES*

We now want to focus in on one area of the software documentation, testing, inspection and qualification process: the software inspection activity. This inspection process includes a number of areas: (1) metrics, (2) software inspection training, and (3) formal software inspection.

Fig 4. **NASA SOFTWARE INSPECTION ACTIVITIES**

- **Implementation of requirements.**
- **Review of pseudo code.**
- **Review of mechanics.**
- **Review of data structure.**
- **Code "walk-thru"**
- **V&V**
- **IV&V**

The objectives of formal inspection include: (1) removing defects as early as possible in the development process, (2) having a structured, well-defined review process for finding and fixing defects, (3) generating metrics and checklists used to improve quality, (4) following total quality management (TQM) techniques--working together as a team and (5) having responsibility for a work product shared by author's peers.

Fig 6. **FORMAL INSPECTION OVERVIEW**

- **Objective to remove defects as early as possible in the development process.**
- **Structured, well-defined review process for finding and fixing defects.**
  - **Conducted by small teams of peers with assigned roles.**
  - **Each participant has vested interest in work product.**
  - **Held within development phases on completed portions of engineering products.**
- **Metrics & checklists used to improve quality.**
- **Responsibility for work product shared by author's peers.**

To achieve these objectives, specifications must be reviewable and formally analyzable. They also must be usable by both the designers and by assurance and safety engineers. Further the specifications must support completeness and robustness checks and they must support the generation of mission test data.

### 3.1. Formal Design Requirements and Inspections

The objective of the inspection process is to remove defects at the earliest possible point in the product development lifecycle. The product can be a document, a process, software or a design. Inspection topics include requirements, design requirements, detailed design requirements, source code, test plans, procedures, manuals' standards and plans.

Inspection is a very structured process that requires each member of a team to have a real interest in the software product. They are involved because of their technical expertise with the product. The inspection should be considered a tool to help the author identify and correct problems as early as possible in the development process. Inspectors should not

be viewed as critics whose only job is to find fault. This inspection should help develop a team environment emphasizing that everybody is involved in developing a high quality product.

Metrics (e.g., minor errors discovered, major errors discovered) generated during this process are used to monitor the type of software defects discovered and to help prevent their reoccurrence.

3.2. Process Overview

Staff, procedures, development time and training are applied to a developing software product to improve its quality. The formal seven step program for inspection includes:

The *planning phase* where organizing for the inspection takes place.

The *training phase* where team members are given background and details for the inspection activity.

The *preparation phase* where individual inspectors review the work prior to the joint inspection meeting.

The *inspection meeting* where the team identifies, classifies and records defects.

The *"third hour"* (cause phase) where the programmers participate in off-line discussions to get help with the defects.

The *rework phase (corrective action)* where the programmers correct the defects.

The *follow-up phase* where the revisions are reviewed and verified by the team.

3.3. Roles

Fig 7. **ROLES IN FORMAL INSPECTION**

| | |
|---|---|
| •Moderator | Coordinates & conducts the inspection process. |
| •Author | Produces the work product and performs rework. |
| •Reader | Presents the work product to the inspection team during the inspection meeting. |
| •Recorder | Documents defects identified during the inspection meeting. |
| •Inspector | Identifies defects in the work product. |

Each person who participates in the inspection takes on various tasks. The *moderator* coordinates the inspection process, chairs the inspection meetings and makes sure the inspection process is carried out.

The *reader* presents the work product to the inspection team during the meeting. The reader does this instead of the programmer (*author*).

The *recorder* documents all the defects, open issues and action items that are brought forward during the meeting.

The job of *inspector* is the responsibility of every person in the meeting. Each person helps to identify and evaluate defects.

3.4. Development Process Benefits.

Some of the benefits of this inspection process for the overall software development process include:

Fig 8. **BENEFITS OF FORMAL INSPECTION FOR SOFTWARE DEVELOPMENT**

- **Improves quality and gives cost savings through early fault detection and correction.**
- **Provides a technically correct base for the following phases of development.**
- **Contributes to project tracking.**
- **Improve communication between developers.**
- **Aids in the project education of personnel.**
- **Provides structure for in-process reviews.**

This inspection process also benefits the software developer in a number of ways:

Fig 9. **BENEFITS OF FORMAL INSPECTION FOR DEVELOPERS**

- **Provides technical support during product development.**
- **Reduces repetition of defects through early detection.**
- **Identifies missing elements in work product (according to data kept by the Jet Propulsion Lab, 60% of all defects are missing requirements).**
- **Provides team development support environment.**
- **Provides project training and expands expertise across development phases.**

Some of the benefits of this inspection process are as follows:

(1) The number of defects made by the author is reduced since defects are identified early in the product life cycle.

(2) Omissions in the requirements are identified efficiently by this process.

(3) The inspection team supports the programmer with constructive criticism and guidance rather than tearing down software in open, public project design reviews.

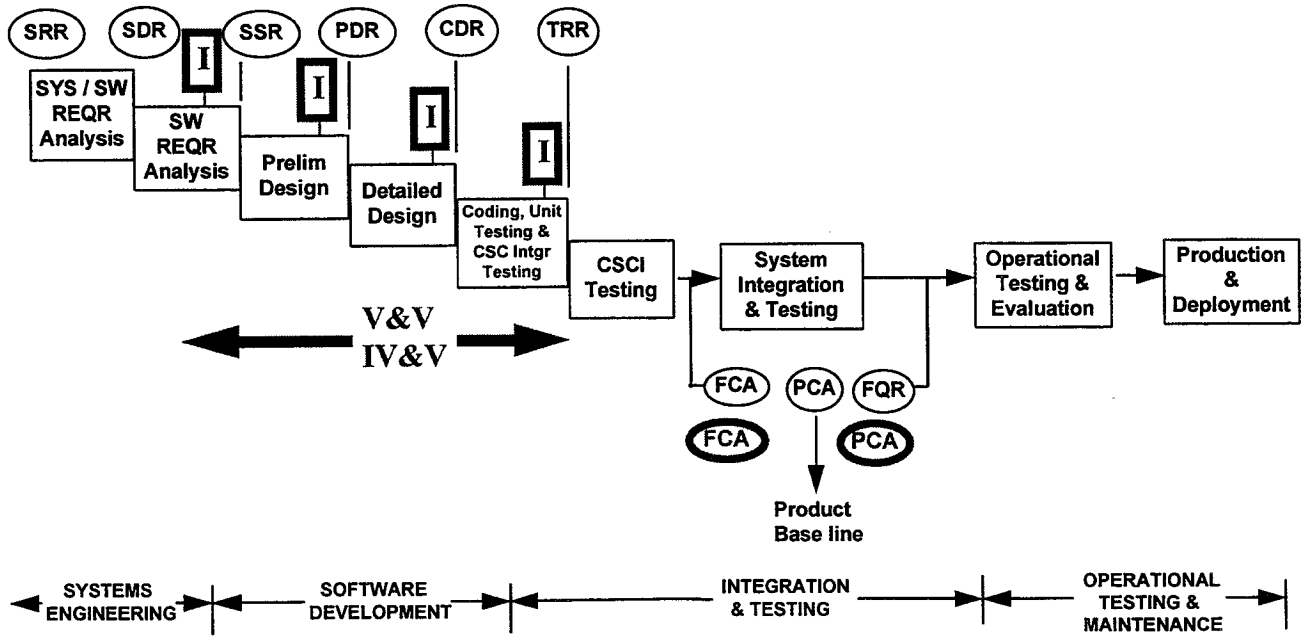
(4) The inspection process benefits the entire team because they benefit from lessons learned and mistakes of others in a constructive atmosphere.

(5) Improved project tracking is implemented with the inspection milestones imbedded in the project.

(6) The inspection process helps bring together project persons from varied backgrounds and the resultant communication helps teamwork and improves understanding of the overall project.

(7) New members of the software development team are trained by working with the senior team members.

**Figure 5.0 -- Flow Chart for the Software Development Process**




The waterfall flow chart of the software development process (Based on phases in DOD-STD-2167A, Defense System Software Development). The acronyms are as follows:

I = Software Inspections
V&V = Verification and Validation Activity
IV&V = Independent Verification & Validation Activity
CSCI = Computer Software (SW) Configuration Item--(Major SW Program)
CSU = Computer SW Unit--(Program Module)

SRR = System Requirements Review
SDR = System Design Review
SSR = Software Specification Review
PDR = Preliminary Design Review
CDR = Critical Design Review
TRR = Test Readiness Review
FCA = Functional Configuration Audit
PCA = Physical Configuration Audit
FQR = Formal Qualifications Review

3.5. Basic Rules of Inspection.

There are a number of basic rules that need to be followed if the software inspection process is to be effective.

Fig 10. **BASIC RULES FOR INSPECTION**

- **Inspections are carried out at a number of points inside designated phases of the software life cycle and compliment major milestone reviews.**
- **Inspections are carried out by peers representing the areas of the life cycle affected by the material being inspected. Everyone participating should have a vested interest in the work product.**
- **Management is not present during inspections. Inspections are not to be used as a tool to evaluate workers.**
- **Inspections are led by a TRAINED Moderator.**
- **TRAINED inspectors have assigned roles.**

These include:

(1) Inspections are in-process reviews conducted during the development of a product in contrast to milestone reviews conducted between development phases.

(2) Inspections are conducted by a small peer team. Each member has a special interest in the project success.

(3) Managers are not involved in the inspection and the results of the inspection are not used as a tool to evaluate developers.

Fig 11. **BASIC RULES FOR INSPECTION** (continued)

- **Inspections are carried out in a prescribed series of steps.**
- **Inspection meetings are limited to two hours.**
- **Checklists of questions are used to define the task and to stimulate defect finding.**
- **Material is covered during the inspection meeting within an optimal page rate range which has been found to give maximum error finding ability.**
- **Statistics on the number of defects, the types of defects, and the time expanded by engineers on the inspections are kept.**

(4) The moderator leads the inspection process and must have received formal training to do so.

(5) Each team member is assigned a specific role as well as that of an inspector.

(6) The inspection process is spelled out in detail and no step of the process is left out.

(7) The overall time of the inspection is preset to aid in meeting the schedule.

(8) Checklists are used to help identify defects.

(9) Inspection teams should work to an optimal inspection rate. The object of the meeting is not to cover as many pages as possible but to identify as many defects as possible.

(10) Inspection metrics on defect type, number and time spent on inspections. These metrics are used to improve the development process, the work product and to monitor the inspection process.

3.6. Results of Software Inspections

Inspections are a cost saving since fixing defects early in the software development cycle is less costly than removing them later.

| | Work Hours Needed to Fix a Defect |
|---|---|
| Formal Inspections | 0.7 |
| Testing | 5 to 18 |

**It is less expensive to fix defects early in the Life Cycle rather than waiting for test!**

Note: Inspection Information is a lab wide average derived from JPL's Formal Inspection Data Base

Fig 12. **Resource Hours Per Defect**

Further the training provided to the team members in the bug identification and removal process is a valuable development tool.

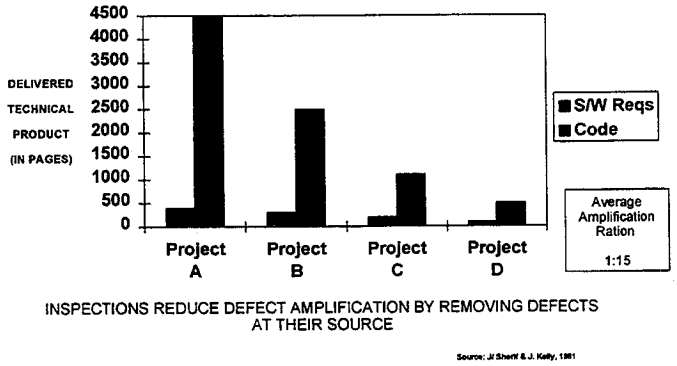



Fig 13. **Amplification Of Requirements Into Source Code**

To fix a defect found with formal inspections costs less than one hour each on the average. To fix a defect found in software test typically has cost from 5 to 18 hours. Defects also tend to amplify. One defect in requirements or design may impact multiple lines of code. A small study conducted by the Jet Propulsion Laboratory (JPL) found an amplification rate of 1 to 15.This means that one defect in the requirements impacts 15 source line of code (SLOC). [1]

- **Developed by IBM Federal Systems Division, Houston.**
- **Inspections applied from 1982 to 1985 on Requirements, Design, Code, Test Plans, Specifications, Procedures.**
- **During this period, operational defect rate was reduced from:**

**2.25 to 0.08 Defects/KLOC**

**This is one of the best examples of Quality improvement resulting from inspections.**

Source: Kolkhorst, 1988

(tz)

Fig 14. **Inspection Experience -- Shuttle Software**

Inspections were used at IBM Federal Systems to develop software for the Space Shuttle. The original defect rate of 2.25 defects per thousand lines of code was unacceptable. Over a three year period, inspections were applied on requirements, design, code and test plans, specifications and procedures. The goal for this effort was 0.2 defects/ thousand lines of code (KLOC). With inspections, the project was able to surpass the goal and reach a defect rate of 0.08 defects/KLOC.

Fig 15. **QUALITY AND COST BENEFITS OF FORMAL INSPECTION**

- **Eliminating defects early - at their source.**
- **Reducing amplification of defects.**
- **Improving software development efficiency.**
- **Improving developer efficiency.**

One of the most essential lessons learned from initial implementation of the inspection process is that all inspection participants require some type of training. Everyone needs to understand the purpose and focus of inspections and the resources required to support the process. Adequate time has to be provided for inspections in the software development process. Furthermore use of metrics from inspections provides an excellent basis for monitoring both the inspection and development process and as a means to evaluate process improvements.

Fig 16. **FORMAL INSPECTION REQUIRE PROJECTS TO HAVE THE FOLLOWING:**

- **An established development life cycle.**
- **An established set of documents produced during the phases of the life cycle.**
- **Software development standards.**
- **Programming standards**

Fig 17. **ADDITIONAL BENEFITS TO THE PROJECT**

- **NASA Software Assurance Standard NASA-STD-2201-93 Requirement:**

**"Software verification and validation activities shall be performed during each phase of the software life cycle and shall include formal inspections"**

Fig 18. **IN SUMMARY**

- **Formal inspections can be used with any development methodology because no matter which development process or lifecycle is used, products are being produced which can be inspected.**
- **Formal inspections are applied during the development of work products. They are a compliment to milestone or formal reviews and are not intended to replace them.**
- **Formal inspections are recommended by the NASA Software Assurance standard and can be applied to the work products called out in the NASA Software Documentation Standard.**

*4. ADDITIONAL RECOMMENDATIONS:*

On the basis of an evaluation of Space Shuttle software development process, the following recommendations were made. [2]

Fig 19. **ADDITIONAL SOFTWARE RECOMMENDATIONS FOR MAJOR PROJECTS**

- **V&V inspection recommendations.**
- **Sufficient personnel.**
- **Standards & procedures applied to all contractors.**
- **Visibility of potential software problems.**
- **Policies & guidelines.**
- **Sufficient resources.**
- **Lessons learned.**
- **Information responsibility.**

(1) *V&V inspections* by contractors should pay close attention to off-nominal cases (crew/ground error, hardware failure, software error conditions). V&V inspection should also focus on verifying consistency between levels of descriptions for modules and verify consistency between module requirements and the design platform. V&V should also assure correctness with respect to the hardware and software platforms. Real independence of IV&V should also be maintained.

(2) Have *sufficient personnel* in system reliability and quality assurance (SR&QA) to support software-related activities and provide sufficient oversight and evaluation of software development activities by the individual SR&QA offices.

(3) Provide for multiple centers on the same program having and enforcing the *same standards & procedures*. Consistent software development coding guidelines should be provided to contractors.

(4) Provide *visibility for potential software problems* by defining detailed procedures to report software reliability, QA or safety problems to the program-level organization.

(5) Provide accepted *policies and guidelines* for development and implementation of software V&V, IV&V, assurance and safety. This should also include a well-documented maintenance and upgrade process.

(6) Provide *sufficient resources*, personnel and expertise to developing the required standards. Also provide sufficient resources, manpower and authority to compel development contractors to provide sufficient information for verification that proper procedures are followed.

(7) Capture *lessons learned* (as mentioned earlier) in the development, maintenance, and assurance of software to be used by other programs. [3,4]

(8) *Precisely identify the information* that each development and oversight contractor is responsible for making available to the community as a whole. Put in place mechanisms necessary to ensure that programs are given all information needed to make intelligent implementations of software oversight functions.

*5. CONCLUSIONS*

The overall software design process will be improved by carefully constructing initial documentation to generate real and usable requirements. Requirements must be capable of being verified by inspection and test.

Fig 20. **FINAL REMARKS**

- **Formal processes, good documentation, real adherence to documentation and standards, applying recommendations of reviewers and taking to heart the software axioms presented will greatly improve the software design and development process.**

These software product assurance activities including formal inspection, production quality metrics, software inspection training, code "walk-through," V&V and IV&V which in turn, are making NASA projects more successful.

*6. REFERENCES*


[1]C. Miller, *JPL Formal Inspection Database Report*, 1993.

[2] Space Shuttle Flight Software Development Processes, NASA.

[3] RAMS Advisory Board, Reliability Preferred Practices for Design and Test, NASA TM 4322, 1991.

[4] RAMS Advisory Board, Recommended Techniques for Effective Maintainability, NASA TM 4628, 1994.

# REPORT DOCUMENTATION PAGE

*Form Approved*
*OMB No. 0704-0188*



| 1. AGENCY USE ONLY (*Leave blank*) | 2. REPORT DATE | 3. REPORT TYPE AND DATES COVERED |
|---|---|---|
| | February 1997 | Technical Memorandum |

**4. TITLE AND SUBTITLE**

Software Design Improvements
Part 2: Software Quality and the Design and Inspection Process

**5. FUNDING NUMBERS**

WU–323–93–03

**6. AUTHOR(S)**

Vincent R. Lalli, Michael H. Packard, and Tom Ziemianski

**7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)**

National Aeronautics and Space Administration
Lewis Research Center
Cleveland, Ohio 44135–3191

**8. PERFORMING ORGANIZATION REPORT NUMBER**

E–10635

**9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)**

National Aeronautics and Space Administration
Washington, DC 20546–0001

**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**

NASA TM–107402, Part 2
IEEE–155NO897–5000

**11. SUPPLEMENTARY NOTES**

Prepared for The International Symposium on Product Quality & Integrity cosponsored by AIAA, ASQC/RD, ASQC/ED, IEEE/RS, IES, IIE, SAE, SOLE, SRE, and SSS, Philadelphia, Pennsylvania, January 13–16, 1997. Vincent R. Lalli, NASA Lewis Research Center; Michael H. Packard, Raytheon Engineers and Constructors, 2001 Aerospace Parkway, Brook Park, Ohio 44142; Tom Ziemianski, Texas Instruments Inc., Dallas, Texas (work funded by NASA Contract NAS3–26764). Responsible person, Vincent R. Lalli, organization code 0510, (216) 433–2354.

**12a. DISTRIBUTION/AVAILABILITY STATEMENT**

Unclassified - Unlimited
Subject Categories 33, 37, and 38

This publication is available from the NASA Center for AeroSpace Information, (301) 621–0390.

**12b. DISTRIBUTION CODE**

**13. ABSTRACT** ***(Maximum 200 words)***

The application of assurance engineering techniques improves the duration of failure-free performance of software. The totality of features and characteristics of a software product are what determine its ability to satisfy customer needs. Software in safety-critical systems is very important to NASA! We follow the System Safety Working Groups definition for system safety software as: "The optimization of system safety in the design, development, use and maintenance of software and its integration with safety-critical systems in an operational environment." "If it is not safe, say so!" has become our motto. This paper goes over methods that have been used by NASA to make software design improvements by focusing on software quality and the design and inspection process.

| 14. SUBJECT TERMS | 15. NUMBER OF PAGES |
|---|---|
| Software; Reliability; Problems; Safety; Improvement; Inspection; Axions | 08 |
| | 16. PRICE CODE: A02 |

| 17. SECURITY CLASSIFICATION OF REPORT | 18. SECURITY CLASSIFICATION OF THIS PAGE | 19. SECURITY CLASSIFICATION OF ABSTRACT | 20. LIMITATION OF ABSTRACT |
|---|---|---|---|
| Unclassified | Unclassified | Unclassified | |